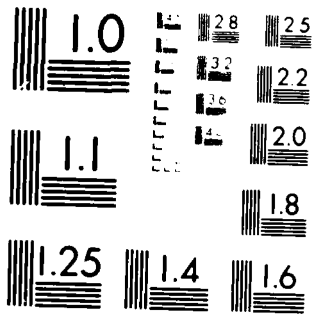
AD-A123 967 MULTILEVEL COMPUTER-AIDED DESIGN OF VLSI DIGITAL SYSTEMS(U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF ELECTRICAL ENGINEERING S W DIRECTOR ET AL. 23 DEC 82 1/1

UNCLASSIFIED ARO-16532.4-EL DAAG29-79-C-0213 F/G 9/5 NL

END
DATE
FILMED
2-83
DTIC


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

AD A123967

ARO 16532.4-EL

12

RITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
EPORT NUMBER: Final	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
ITLE (and Subtitle): Multilevel Computer-Aided Design of VLSI Digital Systems		5. TYPE OF REPORT & PERIOD COVERED: Final 9/79 -10/82
		6. PERFORMING ORG. REPORT NUMBER
UTHOR(s): S.W. Director, D.P. Siewiorek, D.E. Thomas		8. CONTRACT OR GRANT NUMBER(s): DAAG29 79 C 0213
ERFORMING ORGANIZATION NAME AND ADDRESS: Carnegie-Mellon University, Department of Electrical Engineering, 5000 Forbes Avenue; Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
ONTROLLING OFFICE NAME AND ADDRESS: U.S. Army Research Office, Box 12211, Research Triangle Park, NC 27709		12. REPORT DATE: ~~January 6, 1983~~ DECEMBER 23, 1982
		13. NUMBER OF PAGES: 22
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report): Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited

DTIC ELECTE JAN 31 1983 S D B

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official department of the army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer-aided design of VLSI circuits. Mixed level simulation. User-machine interface. Multilevel simulation. Design automation.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report discusses work completed toward the development of a user-oriented, multilevel simulation system for the VLSI circuit designer. Design representations which include several different levels of design have been developed. Two design aids, a mixed circuit and logic level simulator and timing abstraction aid for behavioral simulation have been developed. In addition, an interactive user-machine interface for these design aids has been developed.

DTIC FILE COPY

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

83 01 20 058

Multilevel Computer-Aided Design of VLSI Digital Systems

Final Report

S.W. Director, D.P. Siewiorek and D.E. Thomas

December 23, 1982

U.S. Army Research Office

Contract/DAAG 29 79 C 0213

Carnegie-Mellon University

Table of Contents

List of Figures

Abstract

This report discusses work completed toward the development of a user-oriented, multilevel simulation system for the VLSI circuit designer. Design representations which include several different levels of design have been developed. Two design aids, a mixed circuit and logic level simulator and timing abstraction aid for behavioral simulation have been developed. In addition, an interactive user-machine interface for these design aids has been developed.

1. Introduction

Because of the evolving nature of VLSI design, it is important for designers to consider multiple levels of representation during the design process. Specifically, we may identify the following levels of design:

- The Behavioral Level: at which the behavior of a digital system is described without specifying its structure or implementation. That is, the algorithm of a digital system is described without specifying the state machine or the datapaths that it drives.

- The Functional Structure, or Register-Transfer Logic Level: at which digital system implementation is described in terms of abstract components and their interconnections. A control sequence is often included to specify the evokation order of the abstract components.

- The Structural Logic Level: at which a digital system implementation is described in terms of physically realizable components, such as integrated circuit packages or layout cells, and their interconnections. A control sequence is often included at this level.

- The Gate Level: at which a digital system implementation is described in terms of combinational logic components and their interconnections.

- The Circuit Level: at which a digital system implementation is described in terms of transistors and other primitive electronic components and their interconnections.

- The Mask Level: at which the digital system is described in terms of the geometries of IC masks.

- The Process Level: at which the fabrication process steps which we used to implement the digital system as an integrated circuit is described.

Higher, more abstract, level information usually provides a more global picture of design considerations while the lower, more detailed, levels provide specific technology oriented information. In this report we review work which has been carried out under U.S. Army Research Office Grant DAAG/29/79/0213. This work has resulted in the development of several computer-based tools to aid designers in many phases of the digital system design process. Furthermore, design representations have been defined which include several different levels of abstraction so that our design aids may make use of information at several levels of abstraction. Finally, development of a uniform, interactive, user-machine interface which sits between the designer and the set of computer aids is underway.

More specifically our efforts have resulted in the development of simulation strategies which transcend two adjoining levels of abstraction. These strategies were implemented in the mixed gate level simulator SAMSON [Sakallah 81], and the process simulator FABRICS [Nassif 81]. In addition a technique for extracting timing information from the structured logic level up into the behavioral level [Nestor 81] has been developed. Finally the user-machine interface DELILAH [Slack 82] [Bushnell 82] was developed. A more detailed discussion of this work is given below.

2. Mixed Logic-Circuit Level Simulator

SAMSON (System for Activity-directed Mixed Simulation Of Networks) is a new mixed level simulator which harmoniously combines the seemingly desperate techniques of circuit and logic simulation. Two complementary premises help achieve this harmony. The first is that temporal sparseness [De Man 79], as an attribute of a dynamic system, is level-independent. This leads to the use of event-driven simulation techniques, previously limited to the logic-level, as a common framework for mixed-level simulation. The second premise is that the logic and circuit models are *different* representations of the *same* entity and as such have to be compatible. This allows for the development of a new logic-level model which permits a smooth interface between the circuit and logic parts in a mixed-level network.

SAMSON operates on a network which is modeled as a set of n interconnected subnetworks. Individual subnetworks may be described either at the circuit or at the logic level. Circuit-to-logic and logic-to-circuit signal converters are automatically inserted at the appropriate interfaces. Each subnetwork is considered to be a dynamic entity whose time-domain response can be represented by a sequence of events. Events are associated with those instants of time at which the subnetwork equations have to be solved. For circuit-level subnetworks, such events correspond to instants at which the subnetwork equations are discretized (with an appropriate integration formula) and solved (using an iterative scheme such as Newton-Raphson). For logic-level subnetworks, such events correspond to the instants at which the discrete-valued logic equations (which express the subnetwork outputs in terms of its inputs) are evaluated.

2.1. Temporal Sparseness and Exclusive Simulation of Activity

Large networks tend to be temporally as well as spatially sparse. Spatial sparseness reflects the low level of connectivity among distant parts of a network, whereas temporal sparseness reflects the low level of activity in a network at any given point in time. Both types of sparseness can be advantageously exploited in simulation algorithms in order to increase simulation speed. Spatial sparseness is exploited by applying sparse-matrix methods. Temporal sparseness, on the other hand, is exploited by using event scheduling techniques, such as exclusive simulation of activity (ESA), which have previously been identified with logic simulation. [Ulrich 69].

The application of the ESA principle to a network composed of n circuit-level subnetworks proceeds by allowing each subnetwork to be integrated with an individually tailored sequence of integration steps. This forces the network equations to be temporally decoupled. Previous efforts in event-driven circuit simulation carry out such decoupling in an ad hoc manner, generally by assuming that the coupling between adjacent subnetworks is weak [Chawla 75], [Newton 79]. This approach may lead to erroneous simulation results or even to instability [De Micheli 81]. In SAMSON, the decoupling of the network equations is based on a vigorous model which takes into account the *resulting decoupling errors*. The accuracy of event-driven circuit simulation in SAMSON is, therefore, comparable to that of traditional circuit simulation regardless of the amount of coupling among different subnetworks. The basic steps of the event-driven circuit simulation algorithm in SAMSON at a given instant of time t^* are as follows. Let A denote the set of subnetworks which have pending circuit-level events at t^*, and D denote the set of subnetworks which have events in the future ($t > t^*$). Subnetworks in the set A will be referred to as alert and those in the set D as dormant.

1. Extrapolate the outputs of dormant subnetworks.

2. Discretized the equations of alert subnetworks.

3. Assemble and solve the equations of alert subnetworks.

4. Check the status of dormant subnetworks. If any dormant subnetwork should be alerted, transfer it from set D to set A, discretized its equations and go to Step 3.

5. Calculate, for each alert subnetwork, the truncation errors (TE). If the TE are smaller than a given tolerance, calculate the size of the next step, and schedule a corresponding event in the future.

Assuming that a kth-order integration formula is used to integrate a dormant subnetwork, the extrapolation of each output signal in Step 1 is done using a $(k+1)^{st}$-order polynomial which depends on the previous $(k+1)$ computed solutions as well as the last computed truncation error [Sakallah 81]. Prediction-Based Differentiation [Bokhoven 75] formulae are used for discretization in Step 2. The equations in Step 3 are solved using a Newton-Raphson iteration and Block LU factorization [Sakallah 80]. The status check in Step 4 is equivalent to a truncation error check on the inputs of dormant subnetworks.

2.2. The Logic Simulation Model

The logic-gate model used in SAMSON, is characterized by 4 signal states and a 4-parameter back-end delay operator. Two of the states (H and L) are static and correspond to logical truth and falsehood respectively. The other two states (R and F) are dynamic and correspond to a signal in transition between the static states. The delay parameters are two set-up times (Δ_H and Δ_L) and two transition times (Δ_R and Δ_F) In addition to the pure-delay action characterized by these 4 delay parameters, the delay operator has an inertial component which filters out a small class of narrow spikes.

The most noteworthy feature of the above logic-level model is the absence of an ambiguous or unknown state X commonly employed in logic simulators. By replacing X with the more descriptive R and F transition states many anomalies in existing logic simulators disappear. Furthermore, spikes which are treated as error conditions in simulators using an X state are given the more natural interpretation of incomplete transitions.

2.3. The Mixed-Level Interface

Logic-to-circuit and circuit-to-logic signal converters are automatically inserted by SAMSON at the appropriate interfaces in a mixed-level network. Logic-to-circuit conversion involves the transformation of a 4-state logic signal (a sequence of transitions between the states L, R, H, and F) into a continuous voltage signal. The transformation is accomplished by emitting pre-stored rising and falling voltage waveforms in response to input state transitions into R and F respectively. Spikes are generated if the rising and falling waveforms overlap. Circuit-to-logic conversion is basically a thresholding operation which transforms a continuous voltage waveform into a discrete sequence of state transitions. In addition to thresholding, the slope of the voltage signal in the transition region is monitored to detect spikes and generate appropriate logic state transitions (R to F or F to R).

2.4. The Program Structure

The basic structure of the SAMSON program shown in Figure 2-1. The description of a network to SAMSON consists of two parts: model definitions and model instantiations. Basically, a model is a parameterized multi-terminal structure which serves as a template for creating subnetworks of the same structure but possibly different parameters. Two kinds of models are allowed in SAMSON: logic- and circuit-level. Within each category models can be specified hierarchically, i.e., "larger" models can be constructed from previously defined smaller models. At the lowest level, logic models are specified in terms of boolean equations which compute the value of each output signal in the model as a function of the values of its input signals, and a 4-parameter back-end propagation delay operator. For circuit-level models, the primitive is a 2-terminal branch whose branch relationship may be specified by a user-supplied procedure. The more common linear (resistance, capacitance, voltage source,..., etc.) and nonlinear (diode) branches are built-in. Circuit level models are constructed from interconnections of these primitive branches and any previously defined circuit level models. Examples of model descriptions in SAMSON can be found in [Ulrich 69].

Every subnetwork model, whether it be a logic- or circuit-level model, is preprocessed by SAMSON resulting in a PASCAL[1] solution procedure specific to the model. For a logic-level model the solution procedure evaluates the 4-valued logic function of the input signals for each output terminal. For a circuit-level model, the solution procedure includes PASCAL code for loading the coefficients of the Jacobian matrix, performing LU factorization, and forward and back substitution. These model solution procedures are then compiled and added to a model library.

The actual network description consists of a sequence of subnetwork declarations. Each declaration refers to the name of a model of which the subnetwork is an instance. Models can, of course, be either circuit- or logic-level. This establishes an association between the subnetwork and the model solution procedure. Of course, different subnetworks which are instances of the same model share this solution procedure but maintain separate data structures.

3. Process Simulation

In a true multilevel VLSI simulation system, the designer must be in a position to determine the effects of changes in the process on his design. One approach which has been taken is to *a priori* determine the worst case device model parameter values which are likely to occur due to variations in the process, and then to use a circuit simulator to verify behavior under nominal and worst case conditions. The accuracy of such an evaluation strongly depends on the accuracy of the device model parameters used in the simulation. These model parameters are often determined from measurements made on manufactured devices.

An alternative and more desirable approach is to employ a process simulator which contains physical models of the fabrication steps and device models. Such a simulator may be tuned to a particular process by means of a few typical in-process measurements, and would then determine the joint probability density function (*jpdf*) which describes the distribution of the device model parameters including the correlation coefficients between element parameters. These statistics can

[1]SAMSON is written in PASCAL and currently runs on a VAX-11/780 computer.

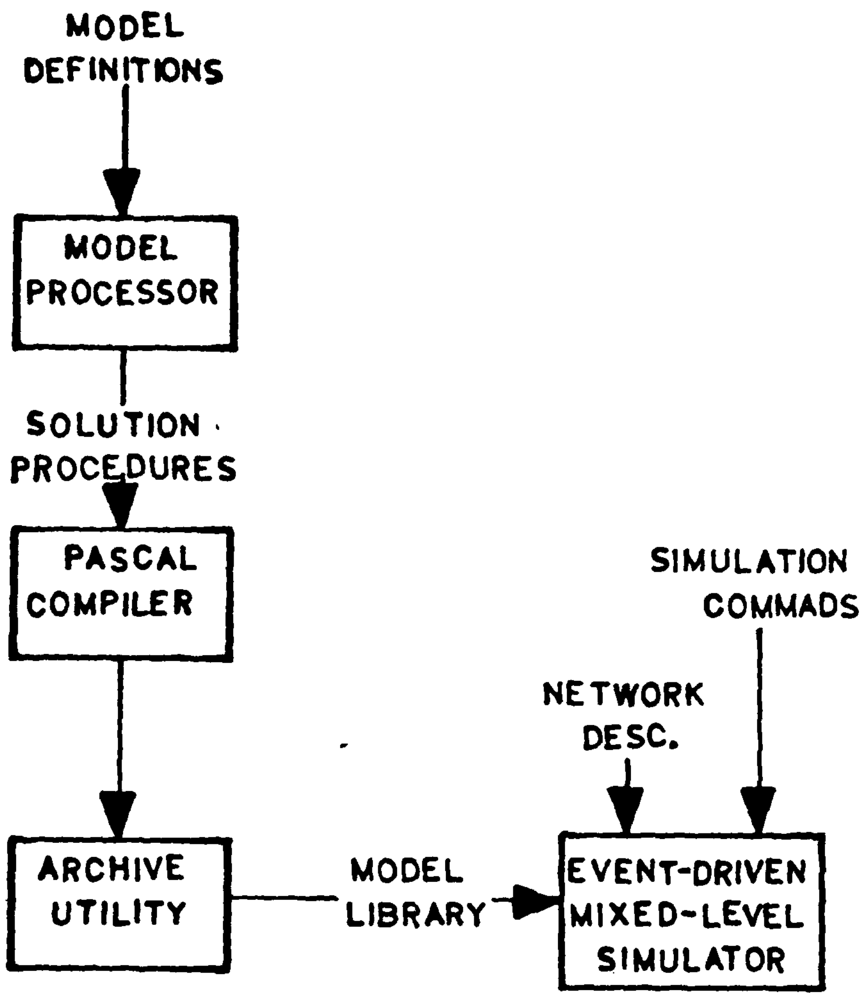


Figure 2-1: The Structure of SAMSON

then be used to generate device parameters suitable for use in a circuit simulator, thereby allowing for accurate yield estimation.

We have begun developing this approach to simulation by combining the circuit simulator SAMSON and the process simulator FABRICS II [Maly 82] [Nassif 81] as shown in Figure 3-1. This software package is flexible and may be used for a variety of design tasks such as providing information about various design alternatives and yield predictions.

3.1. Background

Previous approaches to process simulation (e.g. programs like SUPREM [Dutton 81]) use numerical techniques to solve mathematical models of each fabrication step, thus producing impurity profiles, which can then be input to device simulators (e.g. SEDAN [Dutton 81]) to produce device parameters. Since, for statistical purposes, the process simulator has to generate large data samples, these approaches become very expensive due to the complexity of the models used. Moreover, lack of a direct interface to circuit simulators makes such programs inconvenient to use.

To alleviate these difficulties, FABRICS uses *analytical* models. These models are solutions of the partial differential equations describing the particular fabrication step, under restricted or simplified conditions, that have been found to yield reasonable results. The input t⌝ each model consists of three groups of parameters:

- process parameters (e.g. diffusion times, implanation energies)

- output parameters from previous models

- process disturbances

Process disturbances are statistically independent random variables which represent the fluctuations occurring in the manufacturing steps. These disturbances are modeled as physical parameters, such as duffusivities and segregation coefficients, which probability distribution parameters determined by tuning the model to a particular fabrication process. Similarly, the device models are also analytical functions which use the output from fabrication step models and extracted device layout, to calculate the device parameters. For a more complete description of these models, and of the tuning procedure, see [Nassif 81].

The structure of FABRICS II is shown in Figure 3-2. The program is divided into two major parts, called FAB1 and FAB2, each of which analytic models as described above. This restructuring, as

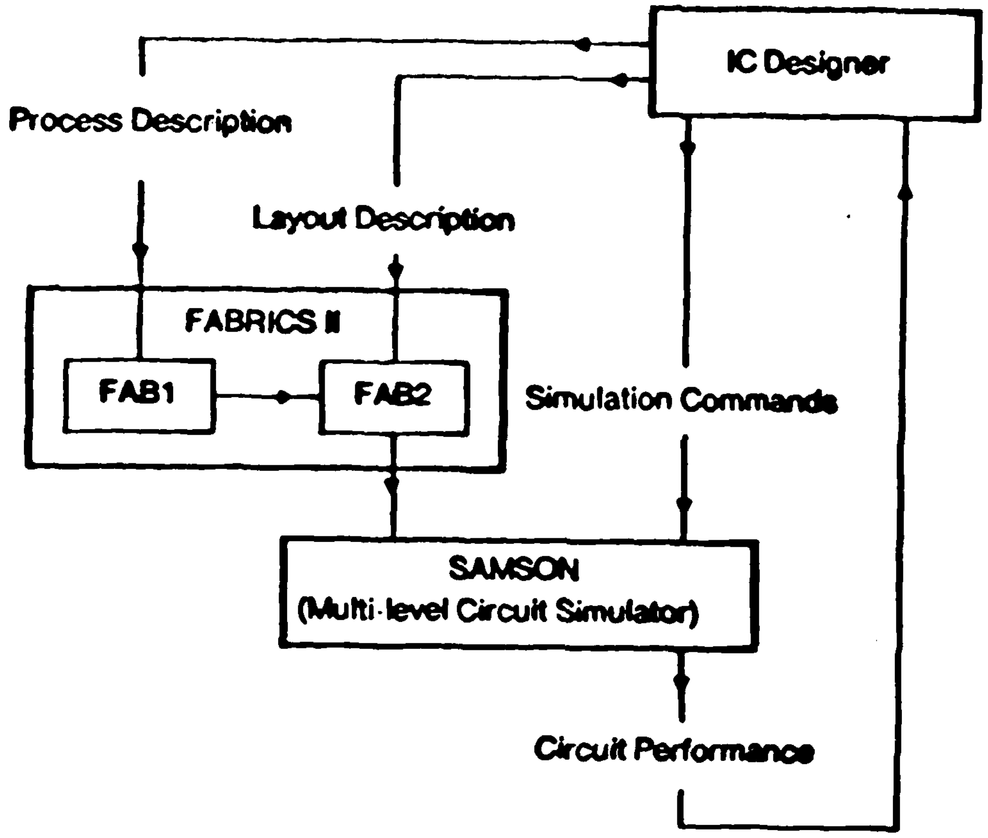


Figure 3-1: The Combined Process-Circuit-Logic Simulation System.

compared to the previous version reported in [Maly 82], makes it easier to modify the program to simulate a variety of different processes, as well as making the addition or modification of fabrication step models transparent to the rest of the simulator.

FAB1 contains models of the manufacturing operations. The input consists of process parameters, process disturbances, and run control parameters. FAB1 simulates the manufacturing operations, such as diffusion and oxidation, producing impurity profiles: which are subsequently processed to produce physical parameters, such as or sheet resistances and junction depths of various impurity layers.

The structure of FAB1, shown in Figure 3-3. allows the simulation of several types of processes, e.g. NMOS, CMOS, Junction-Isolation Bipolar, etc... Each of these processes has a *process supervisor* routine which uses a common set of manufacturing step routines stored in a library. Some of the routines implemented are:

- Diffusion - Predeposition - Drive-in
- Implanation - Annealing - Redistribution
- Oxidation
- Photolithography processes

Currently, a process supervisor for a poly-gate NMOS process has been implemented and a supervisor for a CMOS process is under development.

FAB2 uses the physical parameters generated in FAB1, combined with information about layout dimensions of the various devices, to produce device parameters, such as threshold voltages and betas of MOS transistors, which can be used in a circuit simulator to predict the performance of the circuit.

In the current implementation, where FABRICS is working with the circuit simulator SAMSON [Bokhoven 75], theses parameters are calculated using functions stored in a device library, and used to *fill* a model *template* to be used in SAMSON.

Figure 3-4 shows the structure of FAB2. Note that information about the dimensions of the various devices is derived from a layout description of the chip (e.g. a CIF file) using a circuit extractor. This layout information is divided into two parts: device dimensions, which are used in FAB2 to calculate

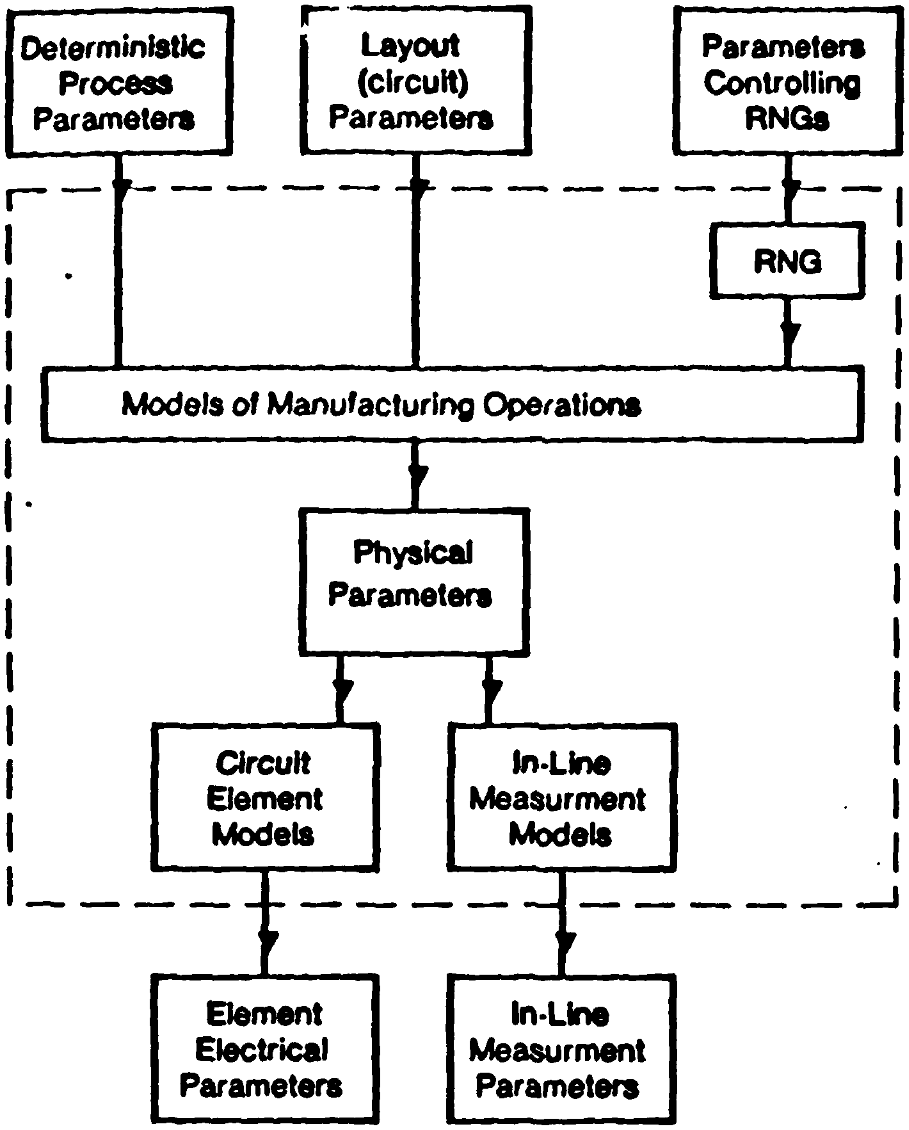

Deterministic Process Parameters
Layout (circuit) Parameters
Parameters Controlling RNGs
RNG
Models of Manufacturing Operations
Physical Parameters
Circuit Element Models
In-Line Measurment Models
Element Electrical Parameters
In-Line Measurment Parameters

Figure 3-2: Structure of FABRICS

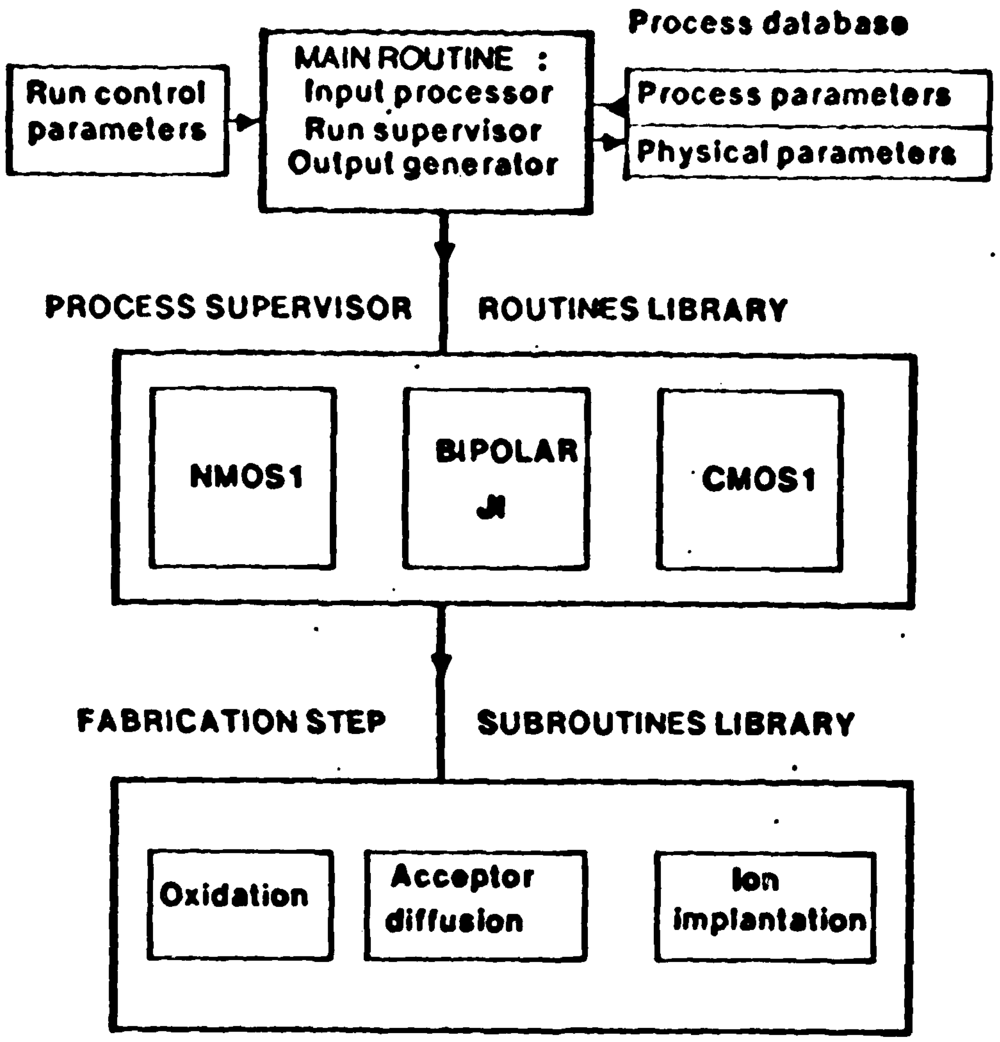


Figure 3-3: Structure of FAB1

device-model parameters, and interconnections, which are modeled in FABRICS to produce R-C type elements, and used in SAMSON to simulate delays between circuit blocks.

4. Multilevel Representation

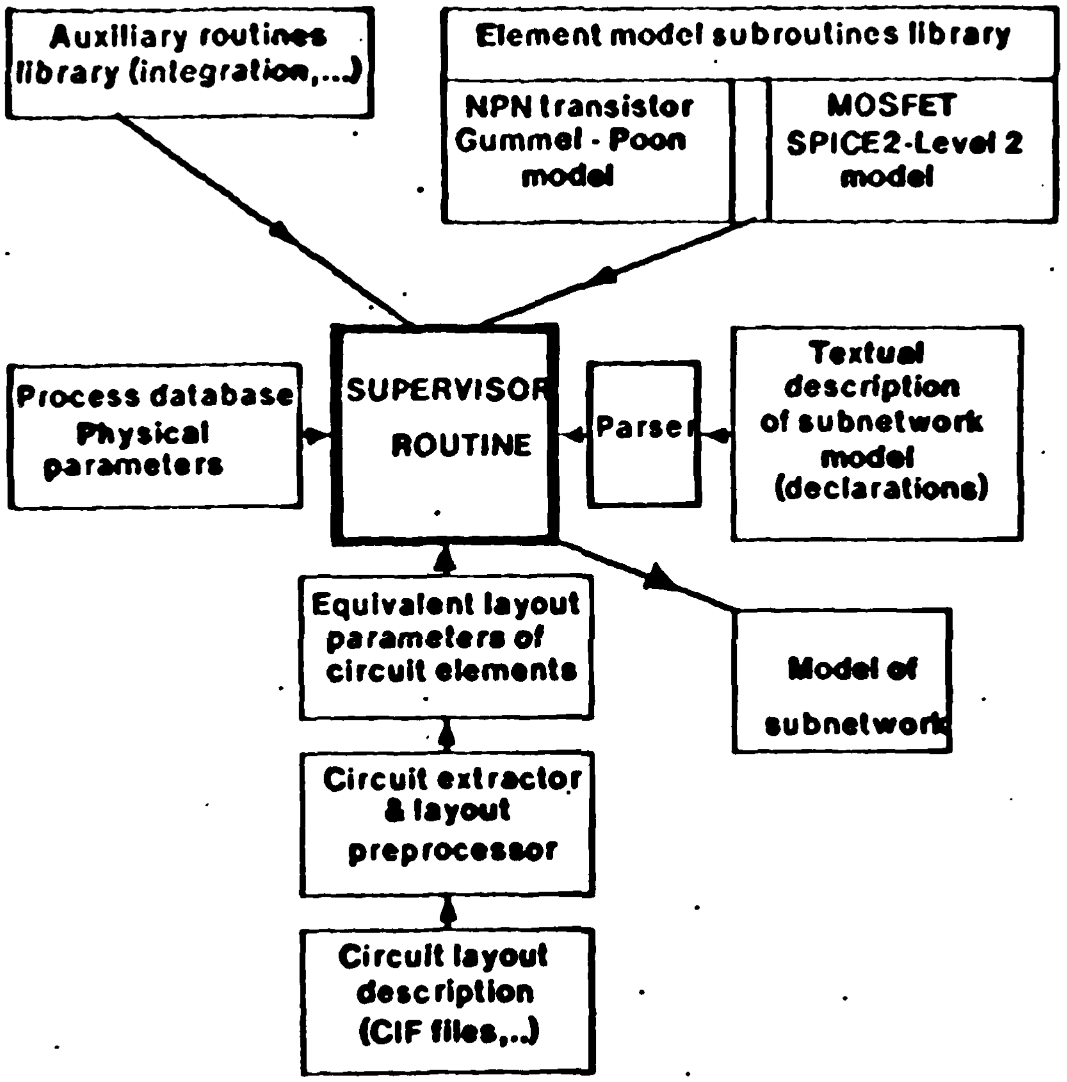
As a first step towards developing a multilevel representation for digital VLSI system, we have developed a test-bed multilevel representation which includes both the behavioral and logical structure levels. This representation was used as a basis from which to abstract timing information from a completed design's microcode and relate it back to the original *behavioral control algorithm.* [Nestor 81] This timing information was then used in the ISPS simulator to demonstrate how a behavioral level simulator could reflect lower level timing data. The advantage of this approach is that an inexpensive simulator can be used to model the performance of a lower level design.

The Timing Abstractor is based on a multilevel representation which is generated from the outputs of the CMU synthesis software [Director 81]. The representation includes *links* which define specific relations between the two kinds of information in the levels: structural and behavioral. With these links the design representations form a single multilevel representation. Figure 4-1 shows this multilevel representational as a hierarchy of design representations and a collection of structural and behavioral links that relate them together.

Structural links relate structural features between each level of representational. Between the behavioral and functional structure description, structural links relate ISPS storage element variables to the corresponding *functional structure register and memory* path graph nodes that implement them. In the present CMU-DA synthesis programs, there is currently a one-to-one correspondence between ISPS register and memory storage elements, and register and memory nodes in the functional structure path graph. Structural links between the behavioral and functional levels therefore link single ISPS storage element variables to single functional structure registers and memories.

Between the functional and logical/physical level design representations, structural links related together several kinds of components, including registers, memories, and operators. Since components in the logical/physical representation are implementations of components in the functional representation, it is generally easy to relate them together.

Behavioral links relate together behavioral features from each level of representation. Behavioral features of a design representation are operations that change the state of the design. In the ISPS



Auxiliary routines library (integration,...)
Element model subroutines library
NPN transistor Gummel - Poon model
MOSFET SPICE2-Level 2 model
Process database Physical parameters
SUPERVISOR ROUTINE
Parser
Textual description of subnetwork model (declarations)
Equivalent layout parameters of circuit elements
Model of subnetwork
Circuit extractor & layout preprocessor
Circuit layout description (CIF files,..)

Figure 3-4: Structure of FAB2

these features are typically ISPS operations. In the functional representation, behavioral features are microsequence operations, which specify how certain data path operations are to be evoked. In the logical/physical structure, behavioral features are microcode operations. Behavioral links relate together the operations in each representation that describe changes into mutually equivalent states at each level of description.

Behavioral links between the behavioral and functional structure descriptions relate ISPS data and control operations to operations in the functional structure microsequence table. These links are one to one, providing that optimizations have not changed the functional structure microsequence operations.

Timing Abstraction extracts the timing information from the microcode of a design and adds it to a corresponding behavioral description. This timing information is proportional to the number of cycles the microcode controller executes between operations that relate to the behavioral description. However, since microcode is almost always optimized, there are cases where some accuracy is lost in timing abstraction.

Microcode optimizations change the order of operations in the microcode, so that the order of operations in the behavioral representation may not always reflect the true ordering of operations in the actual design. In the current timing abstraction algorithm exact timing information is lost for some of the reordered operations. The effect of operation reordering is that timing blocks for these operations are combined. Simulation timing remains accurate at the ends of these timing blocks where the states are mutually equivalent, but the blocks are larger, making the resolution of timing in the simulation more grainy.

Microcode optimizations are constrained to be within straight line segments of microcode [Nagle 80] and do not include control operations. For this reason, the effects of these optimizations on timing abstraction are local. The resulting loss of resolution is local and the resulting loss of resolution is usually small. Furthermore, timing between key operations such as control operations remains accurate, so the inaccuracies due to optimizations are not cumulative.

Use of the timing abstraction algorithm has demonstrated how low level timing detail can be abstracted up to the behavioral level of design for use by system designers. Future research will extend this work through the development of a more extensive multilevel representation and multilevel analysis aids which include the behavioral level of representation.

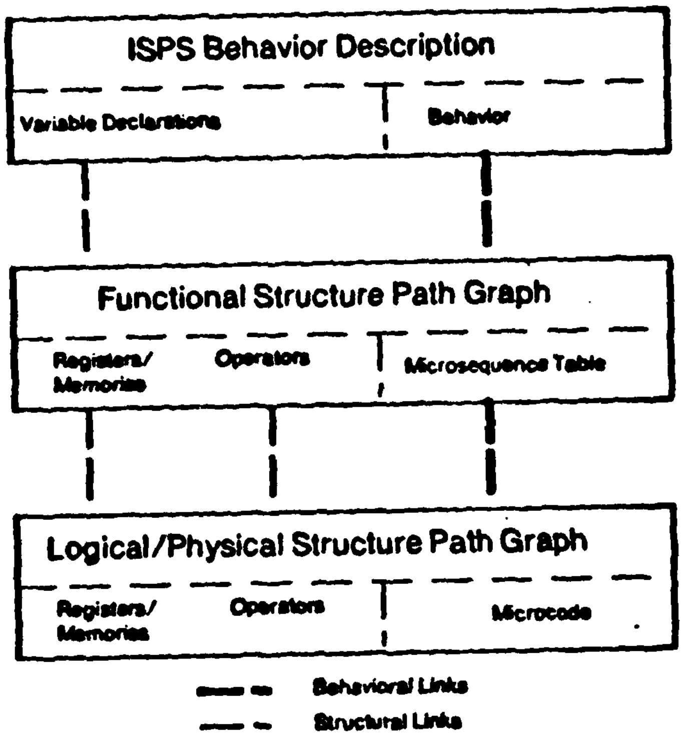


Figure 4-1: The CMU-DA Multilevel Representation

5. The User Interface

This work has been aimed at developing a uniform means of communication between the VLSI designer (called the *user*) and the set of CAD programs (called *application programs*) he will use during the course of design. In general application programs require some form of input and generate some form of output. We define a *user interface* as the program which controls the formal aspects of input and output between an *application program* and the user. Thus, the interface acts as a buffer between the user and the application program. The portion of the interface which decodes the input from the user and passes it to the application program is called the *parser* and the portion of the interface which actually communicates with the user is called the *terminal driver*.

After considering some general attributes of a user-machine interface, we describe a general purpose interface we have developed, called DELILAH.

5.1. Background

Development of applications programs expect in general, three types of responses from users: responses which are selected from a small, well-defined set; responses which set values for, or limits on, a variable; and responses which are arbitrary strings.

When the expected response comes from a small, well-defined list, the user can be shown the entire set and asked to choose one of them. The majority of user interaction falls into this category and will be referred to as a *menu response.*

When the expected response is a value, or set of values, the user needs to know the range from which to choose a response. A reasonable default value may also be provided. We will refer to this type of response as a *range response.*

The third type of response is called a *string response.* As will be seen this type of response can usually be handled as either a menu response or range response.

One of our primary goals in developing a user interface is to create one which can be employed by a variety of application programs, but which interacts in a consistent manner with the user. In order to have a uniform means of interaction we have chosen a menu based scheme thereby assuming that the principle type response expected from the user is a menu response. In order to allow the same basic menu scheme to be usable by a number of different applications programs, we drive the parser by a menu file. This menu file is specific to a given application program and contains a description of the type of user input, and interactive control, expected by a given application program. Note that each application program is accompanied by its own menu file.

We hasten to point out here that while we have assumed a basic menu format, the DELILAH system allows the user to enter command lines (a command line usually consists of a sequence of items found in one or more menus) via the keyboard. Thus, an expert user can skip menu interaction entirely.

5.2. The Menu File

The menu file contains a description of the type of user input and interactive control expected by the application program. This information is characterized in terms of *menu lists, prompts, help messages,* and *menu nodes* where:

- a *menu list* is a list of acceptable user responses and is displayed along the right hand side of the screen;

- a *prompt* is a short message, or question, sent to the user when input is desired and displayed at the beginning of the command line;

- a *help message* is the information presented to the user when an incoherent response is returned, or upon his request.

- a *menu node* is the association of a menu list, a prompt, and a help message; a help message; a menu node must be defined for every point in the application program where user input is expected.

A *menu definition language* has been developed to formalize this information. A complete description of this language can be found in [Slack 82].

5.3. Use of DELILAH

In order to employ DELILAH, the application programmer needs to employ two procedures. *ReadMenu* and *ReadCommand*. The procedure *ReadMenu* designates the menu file to be used by the particular application program. Normally each application program would need only one menu file. The procedure being displayed which is pointed to by the user.

6. Summary

When considered as a whole, the completed research has made progress toward the development of a user-oriented, multilevel simulation system for the VLSI circuit designer. Specifically:

- We have developed a multilevel representation for VLSI which includes the behavioral level of representation and can support multilevel simulation.

- We have developed a mixed gate and logic level simulator.

- We have developed a statistically based process simulator.

- We have developed a timing abstraction algorithm to support system level simulation with detailed timing information.

- We have developed a user-machine interface to conveniently allow use of these systems.

7. References

[Bokhoven 75] Van Bokhoven, W. M. G.
Linear Implicit Differentiation Formulas of Variable Step and Order.
IEEE Trans. Circuits and Systems CAS-22(2):109-115, February, 1975.

[Bushnell 82] M.L. Bushnell.
DELILAH II -- An Enhanced Menu-Driven Input Processor.
Master's thesis, Carnegie-Mellon University, December, 1982.

[Chawla 75] Chawla, Basant R., Gummel, Hermann K. and Kozak, Paul.
MOTIS - An MOS Timing Simulator.
IEEE Transactions on Circuits and Systems CAS-22(12):901-910, Dec, 1975.

[De Man 79] De Man, Hugo J.
Computer-Aided Design for Integrated Circuits: Trying to Bridge the Gap.
IEEE Journal of Solid-State Circuits SC-14(3):613-621, June, 1979.

[De Micheli 81] De Micheli, Giovanni & Sangiovanni-Vincentelli, Alberto.
Numerical Properties of Algorithms for the Timing Analysis of MOS VLSI Circuits.
In R. Boite & P. DeWilde (editor), *Circuit Theory and Design: Proceedings of th 1981 European Conference on Circuit Theory and Design*, pages 387-392. Delft University Press/North Holland Publishing Co., The Hague, The Netherlands, August, 1981.

[Director 81] Director, S. W. and Parker, A. C. and Siewiorek, D. P. and Thomas Jr., D. E.
A Design Methodology and Computer Aids for Digital VLSI Systems.
IEEE Trans. Circuits and Systems CAS-28(7):634-645, July, 1981.

[Dutton 81] Dutton, R.W.
Stanford Overview in VLSI Research.
IEEE Transactions on Circuits and Systems CAS-28(7):654-665, July, 1981.

[Maly 82] Maly, W. and A.J. Strojwas.
Satistical Simulation of the IC Manufacturing Process.
In *IEEE Trans. on CAD.* July, 1982.

[Nagle 80] A. W. Nagle.
Automated Design of Digital-System Control Sequencers from Register-Transfer Specifications.
PhD thesis, Carnegie-Mellon University, 1980.

[Nassif 81] Nassif, S.R.
Extensions to an Integrated Circuit Manufacturing Process Simulator.
Master's thesis, CMU, September, 1981.

[Nestor 81] J. Nestor.
Timing Abstraction Using Multi-Level Design Representation.
Master's thesis, CMU, October, 1981.

[Newton 79] A.R. Newton.
Techniques for the Simulation of Large-Scale Integrated Circuits.
In *IEEE Transactions on Circuits and Systems*, pages 741-749. IEEE/ACM, September, 1979.

[Sakallah 80] K.Sakallah and S.W. Director.
An Activity-Directed Circuit Simulation Algorithm.
In *Proceedings IEEE ICCC*, pages 1032-1035. IEEE, 1980.

[Sakallah 81] K. Sakallah.
Mixed Simulation of Electronic Integrated Circuits.
PhD thesis, Carnegie-Mellon University, November, 1981.

[Slack 82] Slack, Thomas B.
DELILAH: A Menu Directed, Graphics Based, User Environment for CAD Programs.
Master's thesis, Electrical Engineering Department, Carnegie-Mellon University, January, 1982.

[Ulrich 69] Ulrich, E. G.
Exclusive Simulation of Activity in Digital Networks.
CACM 12(2):102-110, February, 1969.

7.1. Technical Papers Published

1. Director, S.W. and Parker, A.C. and Siewiorek, D.P. and Thomas Jr., D.E. "A Design Methodology and Computer Aids for Digital VLSI Systems." *IEEE Trans. Circuits and Systems CAS-28(7*:634-645, July, 1981.

2. Maly, W. and A.J. Strojwas. "Statistical Simulation of the IC Manufacturing Process." In *IEEE Trans. on CAD.* July, 1982.

3. Nassif, S.R., "Extensions to an Integrated Circuit Manufacturing Process Simulator". Master's thesis, CMU, September, 1981.

4. Nassif, S.R., A.J. Strojwas, and S.W. Director. "FABRICS II: A Statistical Simulator of the IC Fabrication process." In *Proc. of the 1982 ICCC."* ICCC, October, 1982.

5. Nestor, J., "Timing Abstraction Using Multi-Level Design Representation." Master's thesis, CMU, October, 1981.

6. Sakallah, K. and S.W. Director. "An Activity-Directed Circuit Simulation Algorithm." In *Proceedings IEEE ICCC,* pages 1032-1035, IEEE, 1980.

7. Sakallah, K. *Mixed Simulation of Electronic Integrated Circuits.* Ph.D. thesis, Carnegie-Mellon University, November, 1981.

8. Slack, Thomas B. "DELILAH: A Menu Directed, Graphics Based, User Environment for CAD Programs." Master's thesis, Electrical Engineering Department, Carnegie-Mellon University, January, 1982.

9. Thomas, D.E. and J.A. Nestor, "Defining and Implementing a Multilevel Design Representation With Simulation Applications", 19th Design Automation Conference.

10. Thomas, D.E. and J.A. Nestor, "Defining and Implementing a Multilevel Design Representation With Simulation Applications", IEEE *Transactions on* Computer Aided Design of Circuits and Systems. 1983.

Participating Personnel

Co Principle Investigators:

Dr. S.W. Director:	Whittaker Professor and head, Department of Electrical Engineering
Dr. D.E. Thomas:	Associate Professor, Department of Electrical Engineering
Dr. D.D. Siewiorek:	Professor of Electrical Engineering and Computer Science

Students Supported

J. Nestor	M.S.E.E. Degree earned: October 1981
D. Giannopoulos	M.S.E.E. Degree earned: July 1981
M. Bushnell*	M.S.E.E. Degree expected: January 1983
K. Sakallah*	Ph.D. Degree earned: October 1982
D. Lapotin*	Ph.D. Degree expected: September 1986
Art Altman*	M.S.E.E. Degree earned: December 1979

Faculty

- Dr. A.C. Parker
- Dr. C. Eastman
- Dr. M.R. Barbacci

*indicates partial support

DAT

FILM

2-8